ARL-CR-0790 ● JAN 2016

US Army Research Laboratory

Modeling Cerebral Vascular Injury

prepared by David Powell
SURVICE Engineering Company
4965 Millennium Drive
Belcamp, MD

under contract FA8075-14-D-0001

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-CR-0790 ● JAN 2016

US Army Research Laboratory

Modeling Cerebral Vascular Injury

prepared by David Powell
SURVICE Engineering Company
4965 Millennium Drive
Belcamp, MD

under contract FA8075-14-D-0001

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2016	Final	February 2015–June 2015

4. TITLE AND SUBTITLE
Modeling Cerebral Vascular Injury

5a. CONTRACT NUMBER
FA8075-14-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
David Powell

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-WMP-B
Aberdeen Proving Ground, MD 21005-5069

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-CR-0790

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Many numerical models for the brain do not include the vascular structures within the brain and thus are incapable of predicting damage to the cerebral vasculature. The presence of the vasculature within the brain produces a reinforcing effect and leads to an increase in the effective stiffness of the combined brain-vascular material. This work developed a code capable of postprocessing an existing head and brain model to determine the deformation in a corresponding cerebral vasculature mesh. With this information, the user can then input the stretches and strains of the vasculature into a damage model of their choice to predict potential vascular injury. The code is capable of using the geometry of the cerebral vasculature to provide information for an updated anisotropic material model, allowing the directional nature of the vessels to inform the material response of the surrounding brain tissue.

15. SUBJECT TERMS
traumatic brain injury, vasculature, injury biomechanics, numerical brain model, anisotropy of brain tissue

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	50	Christopher P Hoppel
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER (Include area code) 410-278-8878

Contents

List of Figures

List of Tables

1. Introduction

Traumatic brain injury (TBI) is a serious concern for the military and the general civilian population. Blast-related TBI has been prevalent in recent military conflicts (Gupta and Przekwas 2013). During Operation Iraqi Freedom, a study of casualties requiring level V care at Walter Reed Army Medical Center reported that 29% of those that were screened had a TBI. Blasts and explosions were the most common causes, accounting for 78% of those found to have a TBI (Traumatic Brain Injury Task Force 2007). Understanding TBI also has significant relevance for the civilian population. Approximately 1.4 million people within the United States sustain a TBI each year. Of that number 50,000 die, 235,000 are hospitalized, and 1.1 million are evaluated, treated and released from emergency departments (Langlois et al. 2006). When one also considers concussions (often called mild TBI) it is possible the largest proportion of patients are not even seen in an emergency department. TBI is even more concerning due to its residual effects. It is estimated that at least 5.3 million Americans, almost 2% of the population, have current long-term or lifelong disabilities as a result of TBI (Defense and Veterans Brain Injury Center 2014).

TBI associated with closed head injuries, also referred to as nonpenetrating head injuries, can be caused by blast, blunt-force impact, or sudden acceleration. In cases such as these, diffuse axonal injury is one particular injury mechanism that has been cited as a signature injury of TBI neural damage (Taber et al. 2006; Gupta and Przekwas 2013). Deformation of the brain tissue can induce misalignment in the cytoskeletal network or axolemma permeability, inducing a cascade of subcellular events culminating in the severance of the axon (Christman et al. 1994; Smith et al. 2003). It is these axon fiber bundles that make up the structural network that allows neurons to communicate with one another. Injury to the axons leads to degraded structural connectivity, which may be responsible for the cognitive deficits that are characteristic of mild, moderate, and severe cases of TBI (Vettel et al. 2010). Concussion, or mild TBI, is thought to be a less severe type of diffuse axonal injury where axons are damaged to a minor extent from stretching. Postmortem studies of brains with concussions have found axonal damage; however, because of other factors, such as restricted blood flow, it is not possible to isolate the cause of this damage and solely link it to concussions.

While substantial work has already been performed by the Soldier Protection Sciences Branch to investigate axonal injury, another aspect of TBI, hemorrhage and other forms of vascular injury, has not been represented within the numerical models. In 2006 a study of head injury sustained by Soldiers in Operation Iraqi Freedom was completed by Dr Rocco Armonda. Of the patients in the study, 82%

suffered blast injury and 14% were injured by a gunshot wound. Of the entire group, injury to the cerebral vasculature was seen in a significant number of patients. The data indicate that 47% had vasospasm, 35% had a pseudoaneurysm, 12% had a subarachnoid hemorrhage, 3.5% had an epidural hematoma, 16% had a subdural hematoma, 11% had an intraventricular hemorrhage, and 14% had mixed hemorrhages (Armonda et al. 2006). In order for our numerical models to better predict the extent of injury to the Solider, a method to include the cerebral vasculature must be implemented. This report details one such approach, which builds upon existing simulation work within the Soldier Protection Sciences Branch, and through a postprocessing algorithm, determines the strains within the cerebral vascular network.

2. Procedure

The following approach to modeling injury to the cerebral vasculature takes advantage of an existing head and brain model developed within the Soldier Protection Sciences Branch (Kraft et al. 2010; Kraft and Dagro 2011; Kraft et al. 2012; Dagro et al. 2013; McKee et al. 2013). The algorithm will use the results from those simulations to calculate the deformation and strain within a cerebral vascular network. The intention is to leave the existing head and brain mesh untouched and create an overlapping beam mesh of the cerebral vasculature (Fig. 1). The proposed algorithm would serve as a link between the 2 meshes such that deformation and strain could be mapped from the full finite element simulation of the head and brain onto vascular elements. Thus, the vascular model would not constitute a full finite element simulation on its own but rather a postprocessing of the results from an already existing finite element model. The deformation data mapped onto the vascular network can then be fed into a damage model selected by the user to identify the locations at greatest risk for injury.

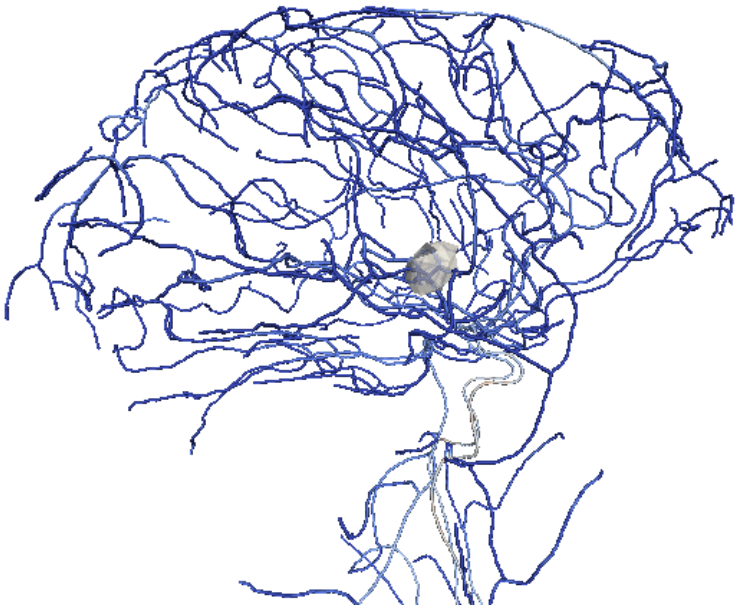

Fig. 1 A 3-dimensional representation of the cerebral vasculature showing larger arteries. Smaller capillaries are not pictured but can also be included.

This approach would require no alternations to the existing simulation. However, once the algorithm has determined the linkages between the 2 meshes, it is possible to use the information regarding the location and orientation of the vascular segments to generate an updated material model for the elements within the brain. The presence of the vasculature running through the brain provides an increased stiffness to its mechanical response. In most simulations this is not directly modeled, instead the stiffness of the brain tissue is increased to account for the vascular presence. Unfortunately, such an approach fails to capture the directional nature of the vascular network. Therefore, by using the location of the vasculature and its orientation a more accurate anisotropic material model can be employed in the initial head and brain simulation. As before, this would still be separate from the vascular model, only with new material properties defined for the elements containing vascular segments.

Thus, this project will provide both a pre- and a postprocessing algorithm built around the linking of a solid element brain mesh with a beam network mesh for the cerebral vasculature. The following subsections will begin by detailing the linking of the 2 independent meshes, the common core of the algorithm, before moving on to the pre- and postprocessing portions of the code.

2.1 Reading Mesh Information and Output Data

To determine the links between the solid element brain mesh and the vascular network mesh, and manipulate the deformation data produced by the finite element

simulation, the proposed algorithm must be able to interface with the files generated by the meshing programs and the results produced by finite element codes such as SIERRA. Reading keyword format input files such as those used in commercial codes like LS-DYNA is relatively straightforward and can be done using the standard input and output libraries for C/C++. However, other finite element codes, including SIERRA, use the EXODUS II file format for both mesh data and output data. The EXODUS II file format was developed at Sandia National Laboratories specifically to store and retrieve data for finite element analyses. To interface with the files in the EXODUS II format, special libraries and commands are required that are not included in standard C/C++ packages.

The files necessary to generate the EXODUS II libraries are available and can be retrieved from sites like SourceForge.net. The user manual is included in Appendix A. A sample input file is provided in Appendix B and alternate format for postprocessing data is provided in Appendix C. A list of useful EXODUS II commands is included in Appendix D.

2.2 Relating Vessel Locations to Brain Mesh

Since the initial locations of the nodes and elements associated with the brain mesh are independent of the location of the nodes and elements associated with the vascular mesh, an algorithm is needed to relate the 2. It is necessary to know which solid (brain) element contains a given beam (vascular) element to calculate the strains in the vascular mesh, and use the orientation of the vasculature to inform the anisotropic response of the surrounding brain tissue. However, because most of the calculations are done in an element reference domain, as opposed to the physical (x, y, z) domain, it is also necessary to relate a given point on a beam element (x, y, z) to the reference coordinates (ξ, η, ζ) for the surrounding solid element as shown in Fig. 2.

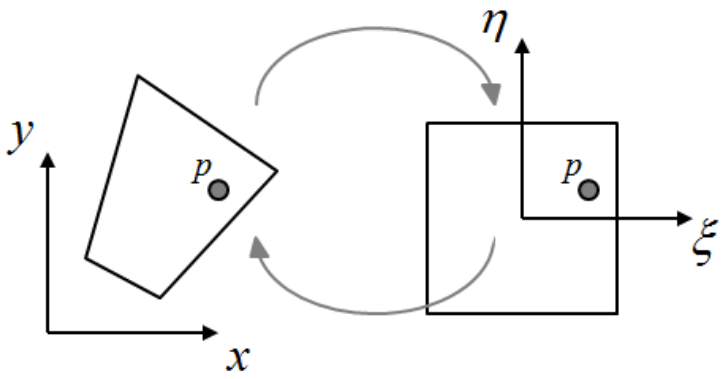

Fig. 2 Mapping from the current domain (x, y, z-space) to the element domain (ξ, η, ζ-space)

Mapping a point from the element domain to the physical domain can be done with the element shape functions, N_a, and the location of the nodes in the physical domain, $\underline{\hat{X}}^a$, using the following equation:

$$\underline{X} = \sum_{a=1}^{NEN} N_a \underline{\hat{X}}^a, \tag{1}$$

where *NEN* is the number of nodes and shapes functions associated with that element. For a typical simulation of the brain and head, the solid elements will be either trilinear hexahedral elements or linear tetrahedral elements. The shape functions for a trilinear hexahedral element are

$$N_a(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_a \xi)(1 + \eta_a \eta)(1 + \zeta_a \zeta), \tag{2}$$

where the parameters for the shape functions are given in Table 1.

Table 1 Hexahedral shape function parameters

a	ξ_a	η_a	ζ_a
1	–1	–1	1
2	1	–1	1
3	1	1	1
4	–1	1	1
5	–1	–1	–1
6	1	–1	–1
7	1	1	–1
8	–1	1	–1

The shape functions for the linear tetrahedral elements are as follows:

$$N_1(\xi,\eta,\zeta) = \xi\ , \tag{3}$$

$$N_2(\xi,\eta,\zeta) = \eta\ , \tag{4}$$

$$N_3(\xi,\eta,\zeta) = \zeta\ , \text{ and} \tag{5}$$

$$N_4(\xi,\eta,\zeta) = 1 - \xi - \eta - \zeta\ . \tag{6}$$

While the shape functions are a convenient way to map from (ξ,η,ζ) to (x, y, z), what is required here is the reverse mapping (x, y, z) to (ξ,η,ζ). While this can be derived for a linear tetrahedral element, a solution cannot be simply written for a general trilinear hexahedral element. Instead we must employ a solution technique such as Newton's method to solve the nonlinear system of equations. Because this approach is computationally intensive, the search algorithm is broken up into an initial coarse search, which is inexpensive to perform, followed by a fine search where Newton's method is employed. For large meshes, the reduction in computational time can be significant.

Prior to initiating the coarse search, the location of centroid, $\underline{X}^c$, for each solid element is calculated

$$\underline{X}^c = \frac{1}{NEN}\sum_{a=1}^{NEN} \underline{\hat{X}}^a\ . \tag{7}$$

Following that, the radius of the smallest sphere containing all points within a given element and centered at the element's centroid is calculated (Fig. 3).

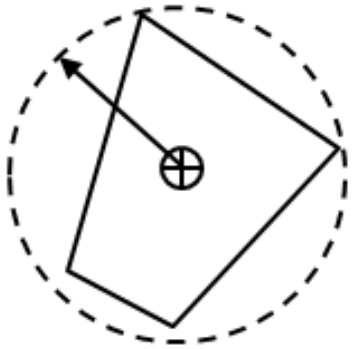

Fig. 3 The centroid of the element and sphere just large enough to contain all of the points within the element

To perform the coarse search, the distance from the centroid of each solid element to a given point in the vascular mesh is calculated. If that distance is less than or equal to the previously calculated radius associated with that solid element, then

the element potentially contains that point and a fine level search is required. If the distance to the point is greater than the radius, then the point cannot fall within the element, and the fine level search is unnecessary.

The fine level search will determine whether the point truly falls within the element, like point *p2* in Fig. 4, or if it is still outside, like point *p1*. To do this we must solve the following system of equations for ξ, η, ζ:

$$\left[\sum_{a=1}^{NEN} N_a(\xi,\eta,\zeta)\underline{\hat{X}}^a\right] - \underline{X} = 0 \quad , \tag{8}$$

where $\underline{\hat{X}}^a$ are the nodal positions for the given element and $\underline{X}$ is the location of the vascular point in the physical domain, both known quantities. To begin we rewrite this as a function, $\underline{f}\left(\underline{\xi}\right) = \left[\sum_{a=1}^{NEN} N_a\left(\underline{\xi}\right)\underline{\hat{X}}^a\right] - \underline{X} = 0$, where the coordinates ξ, η, ζ have been rewritten as the vector $\underline{\xi}$. Without knowing the value of $\underline{\xi}$ that satisfies the equation, we can make an initial guess, $\underline{\xi}^0$, and Taylor expand about it

$$0 = \underline{f}\left(\underline{\xi}^0\right) + \left[\frac{\partial \underline{f}\left(\underline{\xi}^0\right)}{\partial \underline{\xi}}\right]\left(\underline{\xi}^1 - \underline{\xi}^0\right) + \ldots \ . \tag{9}$$

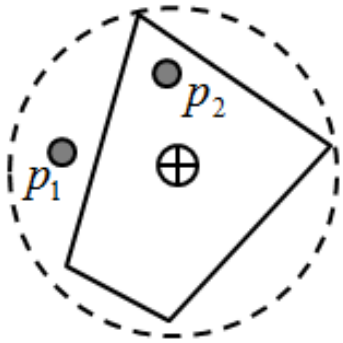

Fig. 4 Points contained within the sphere can also be contained by the element, although it is not necessarily so

Dropping higher order terms, we can solve for $\underline{\xi}^1$. This new solution can be substituted for the previous guess and process repeated, giving us the following equation:

$$\underline{\xi}^{n+1} = \underline{\xi}^n - \left[\frac{\partial \underline{f}\left(\underline{\xi}^n\right)}{\partial \underline{\xi}}\right]^{-1} \cdot \underline{f}\left(\underline{\xi}^n\right). \tag{10}$$

The gradient of $\underline{f}\left(\underline{\xi}\right)$ can be calculated from the shape functions and the nodal positions

$$\frac{\partial \underline{f}\left(\underline{\xi}\right)}{\partial \underline{\xi}} = \left[\sum_{a=1}^{NEN} \underline{\hat{X}}^a \frac{N_a\left(\underline{\xi}\right)}{\partial \underline{\xi}}\right]. \tag{11}$$

This is a 3 × 3 second order tensor and can be easily inverted as long as the elements are convex. The error is defined as

$$error = \left\| \underline{f}\left(\underline{\xi}^n\right) \right\|_2 = \sqrt{\underline{f}\left(\underline{\xi}^n\right) \cdot \underline{f}\left(\underline{\xi}^n\right)}\,. \tag{12}$$

For the given problem, this method typically converges to an error of less than 1×10^{-6} within 1–3 iterations.

The coordinates of the vascular point in the element reference domain can then be compared with the bounds of the element. For the hexahedral element, the coordinates must fall between –1 and 1:

$$-1 \leq (\xi, \eta, \zeta) \leq 1\,. \tag{13}$$

And for the tetrahedral element the coordinates must fall between 0 and 1:

$$0 \leq (\xi, \eta, \zeta) \leq 1. \tag{14}$$

If the coordinates fall within that range, then the point is contained in the current element, and the search can terminate. If any of the coordinates for the vascular point are outside of that range, the point is not within the current element, and the search algorithm moves on to the next element and continues with the coarse search.

2.3 Preprocessing Algorithm

In addition to the goal of linking the solid element head/brain model with the beam element vascular model to predict the strains and potential damage to the vasculature, the vascular model can also be used to improve the accuracy of the head/brain model. The vascular structure within the brain and along its surface provides a stiffening effect and will influence its deformation. In most numerical models the vascular structure is not included. Instead the brain matter and vasculature are merged into a single homogenous material with properties stiffer than white or grey matter would have on their own. Although this approximation is commonly used, it does not take into account the structured nature of the vasculature, and the effect is applied as an isotropic increase in stiffness instead of the more appropriate anisotropic increase in stiffness. The presence of the vasculature can be included as an additional anisotropic term in the constitutive model for the brain tissue based on the direction of the vessels embedded within a

given element. However, let us begin by looking at the material model for the brain prior to the inclusion of the vasculature term.

In many biological tissues, fibers or bundles of cells are aligned in uniform directions. As a result, isotropic materials models are insufficient for capturing the mechanical behavior. For the white matter within the brain, axon bundles form complex fiber tracts as they connect and facilitate communicate between different regions in the brain. These axonal fiber tracts have been reported to be approximately 3 times stiffer than the surrounding matrix material and thus play an important role in the mechanical response of the brain (Arbogast and Margulies 1999). To model the white matter we will use a transversely isotropic hyperelastic material, where the fiber tract directions are determined from diffusion tensor imaging (DTI) data (Kraft and Dagro 2011).

To describe the material model we must first define some basic kinematic concepts. The deformation gradient is defined as

$$\underline{\underline{F}} = \frac{d\underline{x}}{d\underline{X}}, \tag{15}$$

where $\underline{X}$ is the position of a material point in the reference (undeformed) configuration and $\underline{x}$ is the position of the same material point in the current (deformed) configuration. The reference configuration refers to the undeformed physical domain, not the element's reference domain. The ratio of the deformed volume to the undeformed volume is given by the Jacobian, the determinant of the deformation gradient

$$J = det\left(\underline{\underline{F}}\right). \tag{16}$$

It is often beneficial to perform a multiplicative decomposition of $\underline{\underline{F}}$ into volume-changing (dilatational) and volume-preserving (distortional) parts to separate the bulk and the shear response. To accomplish this, a deviatoric deformation gradient in which the volume change is eliminated is defined as

$$\overline{\underline{\underline{F}}} = J^{-1/3}\underline{\underline{F}}. \tag{17}$$

We can then define a modified right Cauchy-Green tensor

$$\overline{\underline{\underline{C}}} = \overline{\underline{\underline{F}}}^{T}\overline{\underline{\underline{F}}}. \tag{18}$$

The modified principle invariants of the right Cauchy-Green deformation tensor are defined as

$$\overline{I}_1 = tr\overline{\underline{\underline{C}}}, \tag{19}$$

$$\bar{I}_2 = \frac{1}{2}\left[\left(tr\underline{\underline{\bar{C}}}\right)^2 - tr\left(\underline{\underline{\bar{C}}}^2\right)\right], \tag{20}$$

and

$$\bar{I}_3 = det\underline{\underline{\bar{C}}} = \left(det\underline{\underline{\bar{F}}}\right)^2 = 1\ . \tag{21}$$

For the modified right Cauchy-Green tensor, because the volume change has been eliminated, the third principle invariant will always be one. To capture the anisotropic nature of the white matter we introduce a unit vector $\underline{a}_0$, assigned using DTI data that describes the direction of the fiber in the undeformed reference configuration. We can then define 2 additional invariants based on the fiber direction

$$\bar{I}_4 = \underline{a}_0 \cdot \underline{\underline{\bar{C}}}\,\underline{a}_0\ , \tag{22}$$

$$\bar{I}_5 = \underline{a}_0 \cdot \underline{\underline{\bar{C}}}^2\,\underline{a}_0\ , \tag{23}$$

where $\bar{I}_4$ and $\bar{I}_5$ arise from the anisotropy and describe the deformation of the fiber family. It should be noted that

$$\bar{I}_4 = \underline{a}_0 \cdot \underline{\underline{\bar{C}}}\,\underline{a}_0 = J^{-2/3}\underline{a} \cdot \underline{a} = J^{-2/3}\lambda^2\ , \tag{24}$$

where $\underline{a} = \underline{\underline{F}}\,\underline{a}_0$ is the direction of the fiber in the current configuration and λ is the stretch in the fiber bundle. Thus, $\bar{I}_4$ will also be useful in evaluating strain based injury criteria for the axon fiber bundles (Kleiven 2007). To take into account the presence of the vasculature we define 2 more invariants

$$\bar{I}_6 = \underline{b}_0 \cdot \underline{\underline{\bar{C}}}\,\underline{b}_0\ , \tag{25}$$

$$\bar{I}_7 = \underline{b}_0 \cdot \underline{\underline{\bar{C}}}^2\,\underline{b}_0\ , \tag{26}$$

where $\underline{b}_0$ is the direction associated with the vasculature in the undeformed configuration.

The strain energy, Ψ, of the transversely isotropic hyperelastic material can be written as a function of the modified principle invariants along with the Jacobian, which describes the change in volume. Assuming that the responses of the fibers and the matrix material are not strongly coupled, we can choose to separate the strain energy into a linear combination of the isotropic and anisotropic components

$$\Psi\left(\bar{I}_1, \bar{I}_2, J, \bar{I}_4, \bar{I}_5\right) = \Psi_{\text{iso}}\left(\bar{I}_1, \bar{I}_2, J\right) + \Psi_{\text{aniso}}\left(\bar{I}_4, \bar{I}_5\right), \tag{27}$$

where $\Psi_{\text{iso}}\left(\bar{I}_1, \bar{I}_2, J\right)$ describes the response of the isotropic matrix and $\Psi_{\text{aniso}}\left(\bar{I}_4, \bar{I}_5\right)$ describes the directional contribution of the reinforcing fiber

bundles. We can then select an appropriate isotropic strain energy function for the matrix component such as a Neo-Hookean or Mooney-Rivlin material model. For the anisotropic response it is suggested to select a Fung material model that includes the exponential behavior characteristic of most soft tissues

$$\Psi_{\text{aniso}}(\bar{I}_4) = k_1 \left[exp\left(k_2(\bar{I}_4 - 1) \right) - \bar{I}_4 \right], \tag{28}$$

where k_1 and k_2 are material constants obtained from a parameter fit to experimental data. This is purely a simple example of a Fung material. Depending on the available data a more complex constitutive model for the fibers may be preferred (Weiss et al. 1996; Holzapfel 2000; Ning et al. 2006).

As with the axonal fibers, the direction of the vessels embedded within a given element can be used to add an anisotropic component to the constitutive equation. In the case where only a single vessel segment is contained in the element, the direction is simply determined from the position of the vessel segment's 2 end points. However, there will often be cases where multiple segments are contained within a single element. In those cases a single average direction must be determined. This can be done by adding the directions of the segments and then normalizing to create a unit vector. However, the ordering of the nodes within a segment can cause the direction of the *j*th segment to be either $\underline{d}_i$ or $-\underline{d}_i$, since

$$\underline{d}_i = \underline{x}_2^j - \underline{x}_1^j, \tag{29}$$

where $\underline{x}_1^j$ and $\underline{x}_2^j$ are the locations of the nodes on either end of the *j*th segment. Two parallel segments of a vessel could potentially cancel each other out depending on the number of the nodes (Fig. 5). To prohibit this from happening, an additional term is included in the vector sum to flip the directions as needed. The average vessel direction is calculated with the following equation:

$$\underline{b}_0 = \frac{\sum_{i=1}^{n} s_i \underline{d}_i}{\left\| \sum_{i=1}^{n} s_i \underline{d}_i \right\|} \tag{30}$$

where s_i is a sign term ($s_i = \pm 1$) described by the equation

$$\begin{aligned} s_i &= 1; && i = 1 \\ s_i &= \frac{\underline{d}_i \cdot \left(\sum_{j=1}^{i-1} s_j \underline{d}_j \right)}{\left| \underline{d}_i \cdot \left(\sum_{j=1}^{i-1} s_j \underline{d}_j \right) \right|}; && i > 1 \,. \end{aligned} \tag{31}$$

For the initial segment $s_1 = 1$, and for all the following segments s_i is used to determine whether or not to flip the *i*th segment based on the sum of the preceding segment directions.

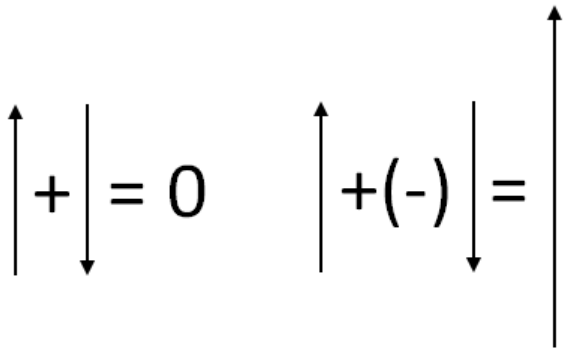

Fig. 5 Depending on order of the nodes in each vessel element, 2 parallel vessel segments could have opposite directions. In the vector sum these will cancel, so the direction segment must therefore be flipped.

Using the average vessel direction, we can calculate our 6th invariant ($\bar{I}_6 = \underline{b}_0 \cdot \underline{\underline{\bar{C}}}\, \underline{b}_0$) and use it to create an additional term within the anisotropic portion on the constitutive equation. As with the response of the fiber tracts, we employ a simple exponential Fung type material model

$$\Psi_{\text{aniso}}(\bar{I}_4, \bar{I}_6) = k_1 \left[exp\left(k_2(\bar{I}_4 - 1)\right) - \bar{I}_4 \right] + k_3 \left[exp\left(k_4(\bar{I}_6 - 1)\right) - \bar{I}_6 \right]. \quad (32)$$

The calculation for the average direction works well when summing multiple segments that are close to parallel. However, when segments are nearly perpendicular an average direction does not have much physical significance. Take the example of an element containing 3 perpendicular fiber segments of equal length. For the sake of simplicity, let those sections be aligned with the x, y, and z axes. The average direction would be the sum of these vectors:

$$\underline{e}_x + \underline{e}_y + \underline{e}_z = \begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix}. \quad (33)$$

And that sum is then normalized to calculate the directional vector associated with that element:

$$\underline{b}_0 = \tfrac{1}{\sqrt{3}} \begin{bmatrix}1\\1\\1\end{bmatrix}. \quad (34)$$

However, because these vectors making up the sum are perpendicular, other vectors would be equally valid as the segment vectors could be defined as $\pm\, \underline{e}_i$ and thus

$$\underline{b}_0 = \pm \tfrac{1}{\sqrt{3}} \begin{bmatrix}-1\\1\\1\end{bmatrix},\ \underline{b}_0 = \pm \tfrac{1}{\sqrt{3}} \begin{bmatrix}1\\-1\\1\end{bmatrix},\ \underline{b}_0 = \pm \tfrac{1}{\sqrt{3}} \begin{bmatrix}1\\1\\-1\end{bmatrix} \quad (35)$$

are also possible solutions. In truth, there should be no preferred direction for this element. So we introduce a scale factor to allow the user to take this into account when creating the material model. This scaling factor is defined as the sum of the absolute values of the dot product of the segments, $\underline{d}_i$, with the previously calculated directional vector, $\underline{b}_0$, divided by the sum of the lengths of the segment vectors

$$f = \frac{\sum_{i=1}^{n} |\underline{b}_0 \cdot \underline{d}_i|}{\sum_{i=1}^{n} \|\underline{d}_i\|} . \tag{36}$$

This allows the length of each directional vector to act as a weight in the sum. The maximum value for the scaling factor, when all of the segments are aligned, is therefore one. For the special case where there are 3 segment vectors of equal length but perpendicular to one another, the scale factor would be $\frac{1}{\sqrt{3}}$ or 0.57735. As a result we expect our scaling factor to have a range of $\frac{1}{\sqrt{3}} \leq f \leq 1$. We can adjust that range by defining a new scaling factor

$$\tilde{f} = \frac{3+\sqrt{3}}{2}\left(\frac{\sum_{i=1}^{n} |\underline{b}_0 \cdot \underline{d}_i|}{\sum_{i=1}^{n} \|\underline{d}_i\|} - \frac{1}{\sqrt{3}}\right), \tag{37}$$

such that $0 \leq \tilde{f} \leq 1$. This scaling factor is a measure of the correlation between fiber directions and can be considered a measure of the anisotropy introduced by the presence of the vasculature. Highlight-aligned vasculature will leave to a scaling factor of close to 1 and have a highly anisotropic effect on the overall response. Conversely, evenly distributed perpendicular vessels will lead to a scaling factor of close to zero and have a more isotropic effect on the overall response. The preprocessing algorithm reports the scaling factor for each element along with the directional vector, $\underline{b}_0$. It is suggested that the updated scaling factor, $\tilde{f}$, be used to weight the anisotropic portion of the constitutive equation that corresponds to the vascular structure

$$\Psi_{\text{aniso}}(\bar{I}_4, \bar{I}_6) = k_1 \left[exp\left(k_2(\bar{I}_4 - 1)\right) - \bar{I}_4 \right] + \tilde{f} k_3 \left[exp\left(k_4(\bar{I}_6 - 1)\right) - \bar{I}_6 \right]. \tag{38}$$

To account for the isotropic stiffening effect of perpendicularly aligned vessels (cases where $\tilde{f}$ is close to zero), the isotropic component of constitutive equation can also be updated

$$\Psi(\bar{I}_1, \bar{I}_2, J, \bar{I}_4, \bar{I}_5) = \left[1 + \kappa^*(1 - \tilde{f})\right]\Psi_{\text{iso}}(\bar{I}_1, \bar{I}_2, J) + \Psi_{\text{aniso}}(\bar{I}_4, \bar{I}_5) . \tag{39}$$

Here the material constant, κ^*, represents the traditional artificial isotropic stiffening of the brain tissue due to the presence of the vessels. As the vessels are more uniformly aligned, that term goes to zero and the anisotropic term takes over. If the vessels are randomly oriented and $\tilde{f}$ approaches zero, the additional

anisotropic term vanishes and isotropic stiffness is increased. This code merely provides a calculation for the average vessel direction and the scale factor. The decision on whether or not to use this data to alter the material model and how it is used within the constitutive equation, is left to the end user.

2.4 Postprocessing Algorithm

To calculate the potential damage in the vascular structure we must be able to map to the deformation from the results of the brain simulation onto the vascular mesh. The algorithm described in Section 2.2 has determined the solid element containing each vessel segment. It has also determined the corresponding coordinates in the element's reference domain. Combining this information with the output data from the simulation of the brain simulation will allow us to determine the strains within the vascular network. Although the initial locations of the vascular nodes are not related to the nodes in the brain mesh, the vascular nodes are treated as material points within the brain mesh. Thus, as the brain mesh deforms, so too do the vascular nodes. It is that deformation which the postprocessing algorithm will determine.

The finite element simulation produces an approximate solution, $\underline{u}^h$, for the displacement, which is a function of the degrees of freedom, $\underline{\hat{u}}^i$, and the shape functions, N_i. Both of these correspond to specific nodes within the descritized finite element mesh.

$$\underline{u}(\underline{X}) \approx \underline{u}^h(\underline{X}) = \sum_{i=1}^{NEN} \underline{\hat{u}}^i N_i(\underline{X}) \quad . \tag{40}$$

While the displacement is a function of the position within the mesh, the $\underline{\hat{u}}^i$'s are constants. For a given element, only the shape functions corresponding to nodes within that element have nonzero values. So, to determine the deformation we only need to sum over the $\underline{\hat{u}}^i$ and N_i values corresponding to nodes within that element. To determine the gradient of the displacement we take advantage of the fact that the $\underline{\hat{u}}^i$'s are constants and move the $\frac{d}{d\underline{X}}$ within the sum. Thus the gradient of the displacement is just the sum of those constants times the gradient of the corresponding shape function at the current coordinate

$$\frac{d\underline{u}}{d\underline{X}} = \sum_{i=1}^{NEN} \underline{\hat{u}}^i \frac{dN_i(\underline{X})}{d\underline{X}} \quad . \tag{41}$$

However, in practice the shape functions are written in terms of the element reference domain's coordinates, so $\widehat{N}_i\left(\underline{\xi}\right)$, and not the physical domain's coordinates, $N_i(\underline{X})$. The gradient of the shape functions with respect to the physical

coordinates can be rewritten as the gradient with respect to the reference domain's coordinates multiplied by the derivative of the reference coordinate with respect to the physical coordinate

$$\frac{dN_i(\underline{X})}{d\underline{X}} = \frac{d\widetilde{N}_i(\underline{\xi})}{d\underline{\xi}} \frac{d\underline{\xi}}{d\underline{X}}. \tag{42}$$

For general trilinear hexahedral elements it is not possible to write a simple expression for $\underline{\xi}$ in terms of $\underline{X}$; however, there is a simple expression for $\underline{X}$ as a function of $\underline{\xi}$

$$\underline{X} = \sum_{i=1}^{NEN} \underline{\hat{X}}^i \widetilde{N}_i\left(\underline{\xi}\right) \tag{43}$$

where $\widetilde{N}_i$ are the same shape functions as before and $\underline{\hat{X}}^i$ are the coordinates of the element's nodes in the physical domain. This can then be easily differentiated to give us

$$\frac{d\underline{X}}{d\underline{\xi}} = \sum_{i=1}^{NEN} \underline{\hat{X}}^i \frac{d\widetilde{N}_i\left(\underline{\xi}\right)}{d\underline{\xi}}. \tag{44}$$

The resulting 3 × 3 tensor can then be inverted to provide the gradient of $\underline{\xi}$ with respect to $\underline{X}$

$$\frac{d\underline{\underline{\xi}}}{d\underline{X}} = \left(\frac{d\underline{X}}{d\underline{\xi}}\right)^{-1}. \tag{45}$$

With the gradient of the descritized displacement determined, we can calculate the strain at the desired location in the brain mesh. This is the strain that the corresponding point in the vascular mesh will experience

$$\underline{\underline{E}} = \frac{1}{2}\left(\frac{d\underline{u}}{d\underline{X}} + \frac{d\underline{u}}{d\underline{X}}^T + \frac{d\underline{u}}{d\underline{X}}^T \cdot \frac{d\underline{u}}{d\underline{X}}\right). \tag{46}$$

This calculation is for the Green-Lagrange strain tensor. While it is beneficial to have the entire strain tensor, we are especially interested in the stretch along the fiber axis. As mentioned in Section 2.3, the stretch along the fiber axis is related to our 6th principal invariant

$$I_6 = \underline{b}_0 \cdot \underline{\underline{C}}\, \underline{b}_0 = \underline{b} \cdot \underline{b} = \lambda^2, \tag{47}$$

where $\underline{b}_0$ is the unit vector describing the direction of the fiber and $\underline{\underline{C}}$ is the right Cauchy-Green tensor. The right Cauchy-Green tensor is related to the strain through the following equation:

$$\underline{\underline{C}} = 2\underline{\underline{E}} + \underline{\underline{I}}\,. \tag{48}$$

The postprocessing algorithm creates a data file containing the element IDs for the vessel segments, the 6 unique components of the Green-Lagrange strain (E_{xx}, E_{yy}, E_{zz}, E_{xy}, E_{yz}, E_{zx}), and the stretch (λ) along the segment axis. These data are repeated for each time step reported by the output of the solid mesh simulation.

3. Example

To demonstrate the capabilities of this code, a sample mesh was generated using trilinear hexahedral elements. This mesh is intended to represent a small section of the larger brain mesh. Figure 6 shows 2 views of this mesh. The mapping from the physical domain to the element's reference domain is greatly simplified in the event of cubic elements or rectangular cuboids. Therefore, the shape of the mesh was chosen so as to force the individual elements to avoid these simpler shapes and better demonstrate the abilities of the code.

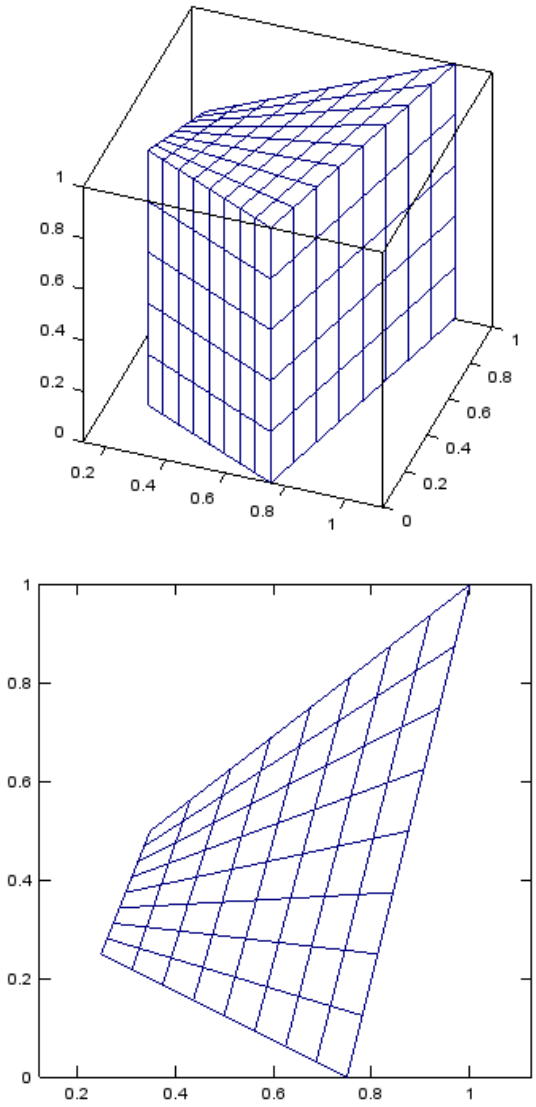

Fig. 6 Solid element mesh representing a portion of the larger mesh

The vascular network mesh can be seen in Fig. 7. This mesh occupies the same physical space as the solid element brain mesh but does not share any of its geometry. The nodes and edges of the 2 meshes do not necessarily coincide. As the vessels move away from their starting point, they branch and alter direction, creating irregular paths through space.

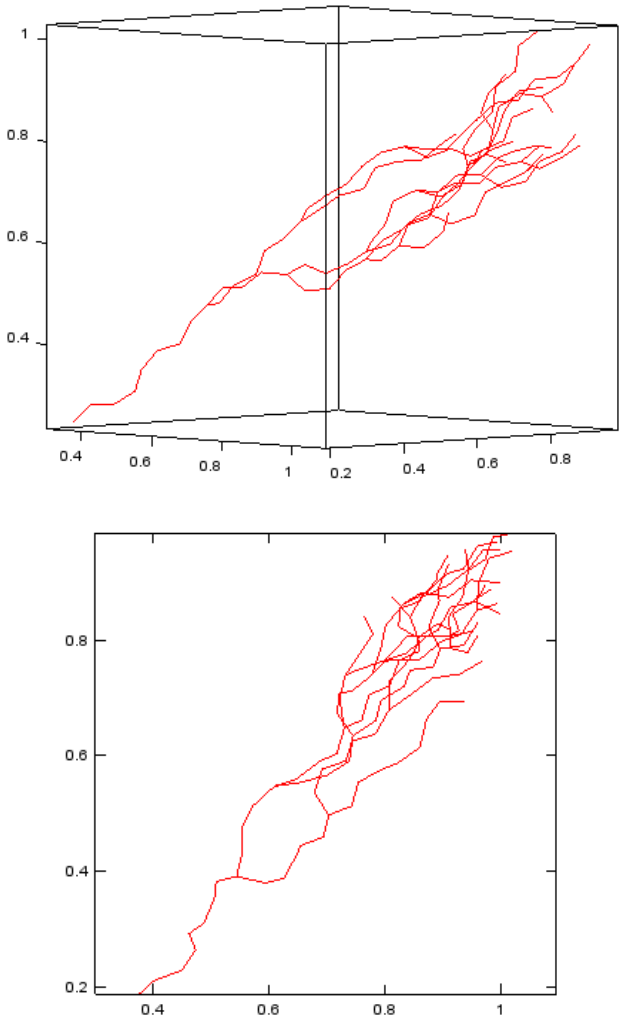

Fig. 7 Vascular structure made up of 1-D elements

The mapping algorithm identifies the elements containing each of the vessel segments and determines the location of the vessel in terms of the element's reference coordinates. Figure 8 shows the vascular network overlaid on the solid element mesh. Using the results of the mapping algorithm, only the elements containing a vessel segment are displayed. For this example the vessels continue on their paths until they have left the solid mesh. Endpoints of the vessels in the image on the right of Fig. 8 can be seen outside of any element. This is not an error in the code, this is because they are outside the boundaries of the solid mesh and thus are not contained within any element. These free endpoints will not be a factor in any of the calculations in either the pre- or postprocessing algorithm.

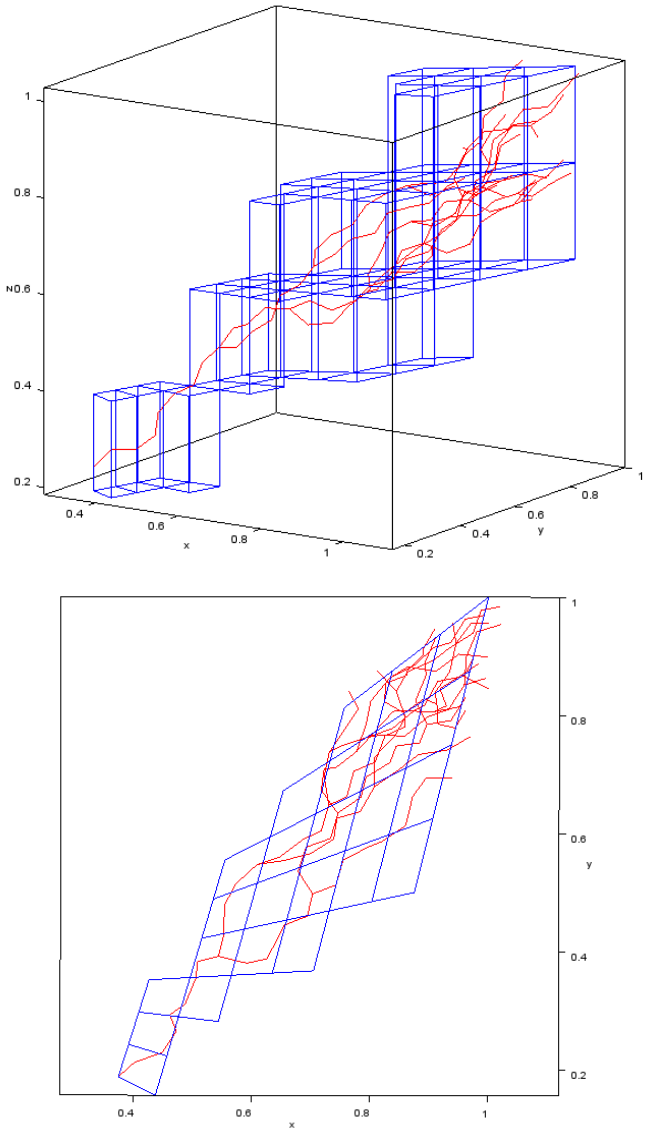

Fig. 8 Vascular structure and solid elements containing vessels

3.1 Preprocessing Results

Using the preprocessing algorithm, every element containing a vessel segment was assigned a directional vector, $\underline{b}_0$, and a scale factor as a measure of the correlation between the directions of the vessel segments. Table 2 shows a sample of the data reported by the preprocessing algorithm. Because of the irregular paths and branching vessels, the scale factor shows a value less than one when multiple segments were present within a single element due to variations in the directions of the individual segments.

Table 2 Preprocessing data for a sample of the solid elements containing vascular segments

Element	X	Y	Z	Scale, $\tilde{f}$
84	0.369179	0.897098	0.771899	1.0
156	0.585461	0.375538	0.718475	0.549186
157	0.455495	0.470486	0.755756	0.791908
164	0.213751	0.930328	0.297994	0.819901
172	0.737664	0.670064	-0.082860	1.0
173	0.933215	0.340099	-0.115940	0.761922
174	0.556493	0.665586	0.497304	0.812248
182	0.303811	0.851047	0.428273	0.389745
190	0.554413	0.826289	-0.099359	1.0
222	0.793040	0.311000	0.523800	1.0
230	0.193325	0.740923	0.643163	0.573547
231	0.581279	0.548160	0.601362	0.753829

3.2 Postprocessing Results

To examine the mapping of strains from the solid mesh to the vessel network, the solid mesh was subjected to a time varying strain that also varied nonlinearly with position. The displacement values are defined at the nodes of the solid mesh only, not the nodes of the vascular mesh. The displacement field defined over the solid mesh was then mapped onto the vascular mesh, and the strains in the vessel segments were calculated. Figure 9 shows a plot of the vascular network with the color of the vessel segments indicating the amount of stretch in each. As expected, the vessel segments in the regions where solid elements have the largest strain also exhibited the largest strain values. However, fiber direction plays a role as well since the stretch along the fiber axis is a function of the fiber's direction

$$\lambda = \sqrt{\underline{b}_0 \cdot \underline{\underline{C}}\, \underline{b}_0} \,. \tag{49}$$

Thus, there is additional variation in the stretch and axial strain within the fibers due to variations in their orientations. Merely calculating the strain within the solid mesh does not provide enough information to make an accurate prediction of the potential for damage in the vasculature; the direction of the vessel segments must be taken into account.

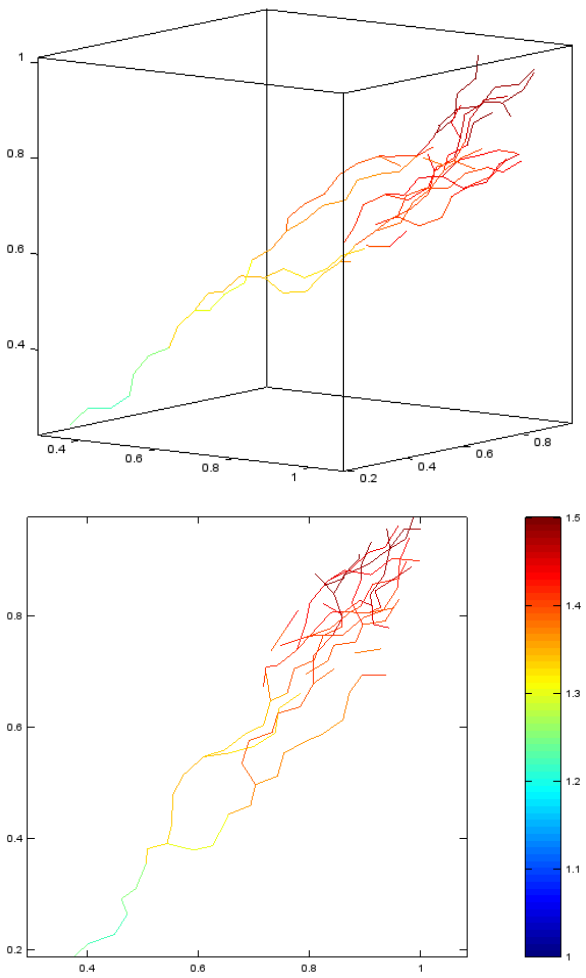

Fig. 9 Axial stretch in the vessels, ranging from 1 (undeformed) to 1.5 (50% elongation)

4. Conclusion

TBI is a serious concern in both the military and civilian population. Even when nonfatal, injury to the brain has the potential for lifelong repercussions. For the Soldier, blast-induced TBI is of particular concern. While significant research has been done on models to predict TBI, many of them have focused on axonal injury and often neglect the possible presence of damage to the cerebral vasculature. During Operation Iraqi Freedom, a study of Soldiers at Walter Reed Army Medical Center suffering TBI found that 47% had vasospasm, 35% had a pseudoaneurysm, 12% had a subarachnoid hemorrhage, 3.5% had an epidural hematoma, 16% had a subdural hematoma, 11% had an intraventricular hemorrhage, and 14% had mixed hemorrhages (Armonda et al. 2006). Injuries such as vasospasm and pseudoaneurysm can lead to further damage to the brain over time. Intracranial hemorrhages and hematomas can be life threatening and often require immediate

intervention. While damage to the vasculature is a common feature of TBI, many numerical models for the brain do not include the vascular structures within the brain and thus are incapable of predicting damage to the cerebral vasculature. This work developed a code capable of postprocessing an existing head and brain model to determine the deformation in a corresponding cerebral vasculature mesh. With this information, the user can then input the stretches and strains of the vasculature into a damage model of their choice to predict potential vascular injury. The code is also capable of using the geometry of the cerebral vasculature to provide information for an updated anisotropic material model, allowing the directional nature of the vessels to inform the material response of the surrounding brain tissue.

This approach is particularly desirable because it builds upon an existing head and brain model currently in use by the Soldier Protection Sciences Branch at the US Army Research Laboratory. Whereas developing an entirely new head model and mesh would be extremely time consuming, this code can be immediately put into use with the existing simulations. Another important feature of this model is that the topography of the brain and vascular meshes can be independent of one another. Other approaches to linking a solid mesh with 1-dimensional (1-D) network often require that the locations of the nodes and edges in the 2 meshes coincide. For the complex geometry of the brain and cerebral vasculature, this requirement would impose severe restrictions on the brain mesh, potentially increasing simulation time or even making effective meshing impossible. This approach places no such restrictions on either mesh and requires no changes to the existing head and brain model. Another important feature of this code is that it can be used to help improve the material model for the brain. The presence of the vasculature within the brain produces a reinforcing effect and leads to an increase in the effective stiffness of the combined brain-vascular material. While many simulations account for this with isotropic increases to the stiffness of the brain's material model, this code can be used to assign directional values to that increased stiffness and lead to a more accurate anisotropic material model. Finally, this code gives the user a great deal of flexibility in how they determine the damage in the vasculature. Since the raw strain and axial stretch data for the vessel segments are reported at each output step, the user can use whatever damage model they deem most appropriate. Although this approach has many strengths, there are some limitations. The model does not currently account for fluid flow within the arteries and the corresponding internal pressure of the vessels. Also, since it uses a 1-D approximation of the vessels, any damage model must be simplified. However, both of these limitations can be addressed through future work.

There are a number of new features that would be desirable if work on the project continues. In regards to the preprocessing algorithm, inclusion of data on the

thickness of the vessels could be used to provide more accurate anisotropic material constants. Larger vessels would have a more pronounced effect on the stiffening of the surrounding brain tissue; therefore the thickness of the segments could be used to provide an additional scaling term for the constitutive equation. Furthermore, a secondary simulation could be created to model the fluid flow within the arteries. Using a pressure gradient to drive the blood flow, and the external pressure induced by a blast wave through the surrounding brain elements, an approximation for the time varying internal pressure of the vasculature could theoretically be created. When this internal pressure calculation is used along with a mapping of the external pressure from the brain, a measure of the compressive stress through the thickness of the vascular walls could be attained. This compressive wall stress is likely an important contributor to and predictor of vasospasm in the arteries. Finally, future work could directly model the largest cerebral vascular structures in the head and brain model, relying on this approach for only the smaller scale vasculature.

5. References

Armonda RA, Bell R, Vo A, Ling G, DeGraba T, Ecklund J, Crandall B, Campbell WW. Wartime traumatic cerebral vasospasm: recent review of combat casualties. Neurosurgery. 2006;59:1215–1225.

Arbogast KB, Margulies SS. A fiber-reinforced composite model of the viscoelastic behavior of the brainstem in shear. Journal of Biomechanics. 1999;32:865–870.

Christman CW, Grady MS, Walker SA, Holloway KL, Povlishock JT. Ultrastructural studies of diffuse axonal injury in humans. J Neurotrauma. 1994;11(2):173–86.

Dagro A, McKee P, Kraft R, Zhang T. A preliminary investigation of traumatically induced axonal injury in a three-dimensional (3-D) finite element model (FEM) of the human head during blast-loading. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2013 Jul. Report No.: ARL-TR-6504.

Defense and Veterans Brain Injury Center. [accessed November 2014] http://www.dbvic.org/clinicalresearch.html.

Gupta RK, Przekwas A. Mathematical models of blast-induced TBI: current status, challenges, and prospects. Frontiers in Neurology. 2013;4:59.

Holzapfel GA. Nonlinear solid mechanics. Chichester (UK): John Wiley and Sons, Ltd; 2000.

Kleiven S. Predictors for traumatic brain injuries evaluated through accident reconstructions. Stapp Car Crash J. 2007;51:81–114.

Kraft RH, Dagro AM. Design and implementation of a numerical technique to inform anisotropic hyperelastic finite element models using diffusion-weighted imaging. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2011 Oct. Report No.: ARL-TR-5796.

Kraft RH, McDowell K, Vettel J. High rate computational brain injury biomechanics. ARL Ballistic Technology Workshop; 2010 May 24–26; Herndon, VA.

Kraft R, McKee PJ, Dagro AM, Grafton S. Combining the finite element method with structural connectome-based analysis for modeling neurotrauma: connectome neurotrauma mechanics. PloS Comp. Bio. 2012;8(8).

Langlois JA, Rutland-Brown W, Thomas KE. Traumatic brain injury in the United States: emergency department visits, hospitalizations, and deaths. Atlanta (GA): Centers for Disease Control and Prevention, National Center for Injury Prevention and Control; 2006.

McKee PJ, Dagro AM, Vindiola M, Vettel JM. Fiber segment–based degradation methods for a finite element–informed structural brain network. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2013. Report No.: ARL-TR-6739.

Ning X, Zhu Q, Lanir Y, Margulies SS. A transversely isotropic viscoelastic constitutive equation for brainstem undergoing finite deformation. Journal of Biomechanical Engineering. 2006;128:925–930.

Smith DH, Meaney DF, Shull WH. Diffuse axonal injury in head trauma. The Journal of Head Trauma Rehabilitation, 2003;18(4):307–316.

Taber K, Warden D, Hurley R. Blast-related traumatic brain injury: what is known? The Journal of Neuropsychiatry and Clinical Neurosciences. 2006;18(2):141–145.

Traumatic Brain Injury Task Force. Report to the surgeon general. np; 2007.

Vettel JM, Bassett D, Kraft R, Grafton S. Physics-based models of brain structure connectivity informed by diffusion-weighted imaging. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2010. Report No.: ARL-RP-0355.

Weiss JA, Maker BN, Govindjee S. Finite element implementation of incompressible, transversely isotropic hyperelasticity. Computer Methods in Applied Mechanics and Engineering. 1996;135:107–128.

INTENTIONALLY LEFT BLANK.

Appendix A. User Manual

To read the mesh files used in SIERRA, this program requires the EXODUS II C and C++ libraries. The files necessary to generate the EXODUS II library can be found at sourceforge.net. These must be compiled and the location of the EXODUS library must be linked when the cerebral vascular injury code is compiled. It is also possible to read in the mesh data using LS-DYNA keyword format; however, the EXODUS II library is still necessary to compile the code. If the user wishes to avoid using the Exodus libraries, it is recommended to create a separate copy of this code with the "include<exodusII.h>" line removed or commented out and all of the EXODUS specific functions in the body of the code.

A-1 General Input File Commands

$ at the beginning of a line denotes a comment. Any information following the $ will be ignored

*PREPROCESS ON/OFF
Determines whether the preprocessing algorithm is run to assign average directions to the brain elements. Acceptable inputs are on, On, ON, off, Off, or OFF. If the command is omitted, the simulation treats it as off.

*POSTPROCESS ON/OFF <format_type> file_name.extension
Determines whether the postprocessing algorithm is run to calculate the strain in the vascular beam elements. Acceptable inputs are on, On, ON, off, Off, or OFF. If the command is omitted, the simulation treats it as off. After the command is the format type for the data and the name of the file containing the finite element deformation data for the solid mesh. The format type is an integer value and must be included. The acceptable values are as follows:
1 – For output in EXODUS II format
2 – For output in text format (for more information see Appendix C)

The file name can include the path to the file if it is not in the current directory.

*NEWTON
$ max_it tol
20 1e-6

Sets parameters for Newton search algorithm.
Max_it – Maximum number of Newton iterations (integer, default = 20)
tol – Tolerance for Newton iteration (double, default 1e-6)

*INCLUDE file_name.extension

Allows for the input file to include other files. Primarily used so that the mesh can be stored in a separate file. Multiple files can be included and those files may also contain the include command. The name of the file and its extension must follow the *INCLUDE command

*END
Denotes the end of an input file. Can be either the end of the main input file or of an include file.

A-2 Mesh Input File Commands

It is possible to read in mesh data either through importing a file in EXODUS II format or using LS-DYNA keyword input format.

*EXODUS_INPUT_SOLID file_name.extension
Read in the mesh file containing the solid elements for the brain/head model. File must be in EXODUS II format.

* EXODUS_INPUT_BEAM file_name.extension
Read in the mesh file containing the beam elements for the vasculature model. File must be in EXODUS II format.

Keyword input files may be used in place of the EXODUS II format files. The following are the commands to input the nodes and elements for either mesh. The simulation will function with either format, you do not need to use both.

```
*NODE
$ NID     X       Y       Z
      1   0.0     0.0     0.0
      2   1.0     0.0     0.0
      3   0.0     1.0     0.0
      4   1.0     1.0     0.0
Etc.
```

NID – Node ID Number (integer > 0): Unique identifier for each node. Numbers must be greater than zero, however numbering does not need to begin at one or be sequential.
X, Y, Z – Reference position for the node (double)

```
*ELEMENT_SOLID
$ EID   type   NID1   NID2   NID3   NID4   NID5   NID6   NID7   NID8
      1      1      1      2     12     11     41     42     48     47
```

```
      2       1       2       3       13      12      42      43      49      48
Etc.
```

EID – Element ID Number (integer > 0): Unique identifier for each element. Numbers must be greater than zero, however numbering does not need to begin at one or be sequential
Type – Element type (integer): Used in LS-DYNA but not only a placeholder here. Some value must be entered but actual value is irrelevant
NID1 – NID8 – Node ID Numbers corresponding to the vertices of the element. A hexahedral element will have 8 NIDs defined. A tetrahedral element will have 4 NIDs defined. Based on the number of NIDs entered, the simulation will automatically determine if a given solid element is a hexahedral or tetrahedral element. All NID numbers must correspond to nodes defined using *NODE

```
*ELEMENT_BEAM
$ Beam Elements that make up the vascular network
$ EID   type   NID1    NID2
      1       1       1       2
      2       1       3       4
Etc.
```

EID – Element ID Number (integer > 0): Unique identifier for each element. Numbers must be greater than zero, however numbering does not need to begin at one or be sequential. Ideally beam elements would not share EIDs with solid elements; however, if they do it should not cause a problem with the simulation.
Type – Element type (integer): Used in LS-DYNA but not only a placeholder here. Some value must be entered but actual value is irrelevant
NID1 – NID2 – Node ID Numbers corresponding to the vertices of the element. All NID numbers must correspond to nodes defined using *NODE

Appendix B. Sample Input File

B-1 Primary Input File

```
*PREPROCESS ON
$
*POSTPROCESS ON 2 post_data.output
$
*NEWTON
$  max_it    tol
       20  1e-6
*INCLUDE Mesh.input
*END
```

B-2 Include File with Mesh

Input file using LS-DYNA keyword format for the mesh. EXODUS II input file can be used in lieu of keyword input (see Appendix A Section A-2).

<Mesh.input>

```
*NODE
5  0.250000  0.250000  0.000000
6  0.500000  0.125000  0.000000
7  0.750000  0.000000  0.000000
8  0.250000  0.375000  0.000000
9  0.562500  0.437500  0.000000
10  0.875000  0.500000  0.000000
11  0.250000  0.500000  0.000000
12  0.625000  0.750000  0.000000
13  1.000000  1.000000  0.000000
14  0.250000  0.250000  0.500000
15  0.500000  0.125000  0.500000
16  0.750000  0.000000  0.500000
17  0.250000  0.375000  0.500000
18  0.562500  0.437500  0.500000
19  0.875000  0.500000  0.500000
20  0.250000  0.500000  0.500000
21  0.625000  0.750000  0.500000
22  1.000000  1.000000  0.500000
23  0.250000  0.250000  1.000000
24  0.500000  0.125000  1.000000
25  0.750000  0.000000  1.000000
26  0.250000  0.375000  1.000000
27  0.562500  0.437500  1.000000
28  0.875000  0.500000  1.000000
29  0.250000  0.500000  1.000000
```

```
30  0.625000  0.750000  1.000000
31  1.000000  1.000000  1.000000
32  0.273756  0.438769  0.210000
33  0.391502  0.420442  0.362500
34  0.509247  0.402116  0.515000
35  0.626992  0.383789  0.667500
36  0.744737  0.365463  0.820000
37  0.273756  0.438769  0.210000
38  0.321508  0.375461  0.387500
39  0.369259  0.312153  0.565000
40  0.417011  0.248845  0.742500
41  0.464762  0.185537  0.920000
*ELEMENT_SOLID
3  1  5  6  9  8  14  15  18  17
4  1  6  7  10  9  15  16  19  18
5  1  8  9  12  11  17  18  21  20
6  1  9  10  13  12  18  19  22  21
7  1  14  15  18  17  23  24  27  26
8  1  15  16  19  18  24  25  28  27
9  1  17  18  21  20  26  27  30  29
10  1  18  19  22  21  27  28  31  30
*ELEMENT_BEAM
11  1  32  33
12  1  33  34
13  1  34  35
14  1  35  36
15  1  37  38
16  1  38  39
17  1  39  40
18  1  40  41
*END
```

B-3 Post-Process Data File

<post_data.output>

```
3
0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
0.000000  0.000000  0.000000
```

0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.100000
0.012500 0.025000 0.000000
0.006250 0.012500 0.000000
0.000000 0.000000 0.000000
0.018750 0.037500 0.000000
0.021875 0.043750 0.000000
0.025000 0.050000 0.000000
0.025000 0.050000 0.000000
0.037500 0.075000 0.000000
0.050000 0.100000 0.000000
0.012500 0.025000 0.050000
0.006250 0.012500 0.050000
0.000000 0.000000 0.050000
0.018750 0.037500 0.050000
0.021875 0.043750 0.050000
0.025000 0.050000 0.050000
0.025000 0.050000 0.050000

0.037500 0.075000 0.050000
0.050000 0.100000 0.050000
0.012500 0.025000 0.100000
0.006250 0.012500 0.100000
0.000000 0.000000 0.100000
0.018750 0.037500 0.100000
0.021875 0.043750 0.100000
0.025000 0.050000 0.100000
0.025000 0.050000 0.100000
0.037500 0.075000 0.100000
0.050000 0.100000 0.100000
0.021938 0.043877 0.021000
0.021022 0.042044 0.036250
0.020106 0.040212 0.051500
0.019189 0.038379 0.066750
0.018273 0.036546 0.082000
0.021938 0.043877 0.021000
0.018773 0.037546 0.038750
0.015608 0.031215 0.056500
0.012442 0.024885 0.074250
0.009277 0.018554 0.092000
0.200000
0.025000 0.050000 0.000000
0.012500 0.025000 0.000000
0.000000 0.000000 0.000000
0.037500 0.075000 0.000000
0.043750 0.087500 0.000000
0.050000 0.100000 0.000000
0.050000 0.100000 0.000000
0.075000 0.150000 0.000000
0.100000 0.200000 0.000000
0.025000 0.050000 0.100000
0.012500 0.025000 0.100000
0.000000 0.000000 0.100000
0.037500 0.075000 0.100000
0.043750 0.087500 0.100000
0.050000 0.100000 0.100000
0.050000 0.100000 0.100000
0.075000 0.150000 0.100000
0.100000 0.200000 0.100000
0.025000 0.050000 0.200000
0.012500 0.025000 0.200000
0.000000 0.000000 0.200000
0.037500 0.075000 0.200000
0.043750 0.087500 0.200000
0.050000 0.100000 0.200000

0.050000 0.100000 0.200000
0.075000 0.150000 0.200000
0.100000 0.200000 0.200000
0.043877 0.087754 0.042000
0.042044 0.084088 0.072500
0.040212 0.080423 0.103000
0.038379 0.076758 0.133500
0.036546 0.073093 0.164000
0.043877 0.087754 0.042000
0.037546 0.075092 0.077500
0.031215 0.062431 0.113000
0.024885 0.049769 0.148500
0.018554 0.037107 0.184000

Appendix C. Alternative Format for Postprocessing Data

Instead of using an EXODUS II file for the deformation data, a text file may be used. That file must match the following format for the algorithm to function. The file will contain only numbers, no keywords are used.

The first line will be a single integer entry indicating the number of output time steps. After that the time of the first output step is reported. Then the following lines include the displacement values of the nodes of the solid mesh in order. All nodes must be reported at each output time step.

<line 1 – Number of output time steps>
<line 2 – Time for output step 1>
<line 3 through number of nodes + 3 – X, Y, and Z displacement values for each node>
<line number of nodes + 4 – Time for output step 2>
<line number of nodes + 5 through 2*(number of nodes) + 5 – X, Y, and Z displacement values for each node>
This pattern continues until all of the output time steps have been reported

For an example of this format, see Appendix B-3.

Appendix D. EXODUS II Commands

The following header line is required at the top of the code to use any of the following EXODUS II commands:

#include “exodusII.h”

D-1 Reading in Mesh Data

Open EXODUS II file
int ex_open(path, mode, comp_ws, io_ws, version);

path – char, file name and path to the EXODUS II file
mode – int, EX_READ or EX_WRITE (predefined constants)
comp_ws – int, word size in bytes (0, 4, or 8)
io_ws – int, word size in bytes (0, 4, or 8)
version – float*, returned EXODUS II database version number

Close EXODUS II file

```
int ex_close (exoid);
```

exoid – int, returned value from ex_open

Read Initialization Parameters
int ex_get_init(exoid, title, num_dim, num_nodes, num_elem, num_elem_blk, num_node_sets, num_side_sets);

exoid – int, returned value from ex_open
title – char*, returned database title
num_dim – int*, returned dimensionality of the database
num_nodes – int*, returned number of nodes
num_elem – int*, returned number of elements
num_elem_blk – int*, returned number of element blocks
num_node_sets – int*, returned number of node sets
num_side_sets – int*, returned number of side sets

Read Nodal Coordinates
int ex_get_coord(exoid, x_coor, y_coor, z_coor);

exoid – int, returned value from ex_open
x_coor – float*, returned x coordinates of the nodes
y_coor – float*, returned y coordinates of the nodes
z_coor – float*, returned z coordinates of the nodes

Read Element Block Parameters
int ex_get_elem_block(exoid, elem_blk_id, elem_type, num_elem_this_blk, num_nodes_per_elem, num_attr);

exoid – int, returned value from ex_open
elem_blk_id – int, ID of current element block
elem_type – char*, returned type of elements in the element block
num_elem_this_blk – int*, returned number of elements in the element block
num_nodes_per_elem – int*, returned number of nodes per element in the element block
num_attr – int*, returned number of attributes per element in the element block

Read Element Block Connectivity
int ex_get_elem_conn(exoid, elem_blk_id, connect);

exoid – int, returned value from ex_open
elem_blk_id – int, element block ID
connect[num_elem_this_blk, num_nodes_per_elem] – int, list of nodes making of the current element

D-2 Reading in Result Data

Read Results Variables Parameters
int ex_get_var_param(exoid, var_type, num_vars);

exoid – int, returned value from ex_open
var_type – char*, type of variable described
"g" or "G" – global variables
"n" or "N" – nodal variables
"e" or "E" – element variables
"m" or "M" – nodeset variables
"s" or "S" – sideset variables
num_vars – int*, returned number of variables stored for specified variable type

Read Variable Names
```
int ex_get_var_names (exoid, var_type, num_vars, var_names[]);
```

exoid – int, returned value from ex_open
var_type – char*, type of variable described

"g" or "G" – global variables
"n" or "N" – nodal variables
"e" or "E" – element variables
"m" or "M" – nodeset variables
"s" or "S" – sideset variables
num_vars – int, number of variables stored for specified variable type
var_names – char**, return array of pointers to num_vars variable names

Read Time Values for a Time Step

```
int ex_get_time (exoid, time_step, time_value);
```

exoid – int, returned value from ex_open
time_step – int, time step number (>= 1)
time_value – float*, returned time at the specified time step

Read Nodal Variables

int ex_get_nodal_var(exoid, int time_step, nodal_var_index, num_nodes, nodal_var_vals);

exoid – int, returned value from ex_open
time_step – int, time step number at which nodal variable values are needed (>= 1)
nodal_var_index – int, index of the desired nodal variable (>= 1)
num_nodes – int, number of nodes
nodal_var_vals – float*, returned array of nodal variables

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA

2 (PDF) DIRECTOR
US ARMY RESEARCH LAB
RDRL CIO LL
IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC
A MALHOTRA

65 (PDF) DIR USARL
RDRL CIH C
A BREUER
B HENZ
M VINDIOLA
RDRL DPW
R COATES
P FROUNFELKER
M TEGTMEYER
RDRL HR
P FRANASZCZUK
K MCDOWELL
RDRL HRS
K OIE
RDRL HRS C
S GORDON
W HAIRSTON
J MCARDLE
A PASSARO
M PETERSON
B LANCE
J VETTEL
RDRL ROP L
F GREGORY
RDRL SLB W
D BOOTHE
N EBERIUS
P GILLICH
A KULAGA
C KENNEDY
W MERMAGEN
RDRL WM
S KARNA
RDRL WML C
L PIEHLER
RDRL WML H
B SCHUSTER
RDRL WMM B
B LOVE
RDRL WMP
S SCHOENFELD
RDRL WMP A
S BILYK
RDRL WMP B
C HOPPEL
W EVANS
A DAGRO
A DILEONARDI
A DWIVEDI
CA GUNNARSSON
Y HUANG
M LYNCH
J MCDONALD
P MCKEE
S SATAPATHY
A SOKOLOW
C WEAVER
T WEERASOORIYA
S WOZNIAK
T ZHANG
K ZIEGLER
RDRL WMP C
DL CASEM
J CLAYTON
T BJERKE
D DANDEKAR
M GREENFIELD
B LEAVY
RDRL WMP D
R DONEY
J RUNYEON
RDRL WMP E
P SWOBODA
RDRL WMP G
R BANTON
N ELDREDGE
S KUKUCK
T PIEHLE
N ZANDER
RDRL WMP F
R GUPTA
R KARGUS
E FIORAVANTE
N GNIAZDOWSKI
R SPINK

INTENTIONALLY LEFT BLANK.